EffiCuts: Optimizing Packet Classification for Memory and Throughput

Authors: Balajee Vamanan, Gwendolyn Voskuilen and T. N. Vijaykumar

Published in: SIGCOMM 2010

Presenter: Guoyao Feng
Packet Classification

- **Goal**
  - Categorize packets by matching it against the highest priority rule

- **Why classify packets?**
  - Firewall / NAT
  - Quality of service
  - Traffic analysis

- **Rule example**

<table>
<thead>
<tr>
<th>Rule ID</th>
<th>Network-layer destination</th>
<th>Network-layer source</th>
<th>Transport-layer destination</th>
<th>Transport-layer protocol</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>128.2.190.69/32</td>
<td>128.2.80.11/32</td>
<td>*</td>
<td>*</td>
<td>Deny</td>
</tr>
<tr>
<td>R2</td>
<td>128.3.3.0/24</td>
<td>128.2.200.157/32</td>
<td>eq www</td>
<td>UDP</td>
<td>Allow</td>
</tr>
</tbody>
</table>
Challenges Facing Modern Classifiers

- Classifiers growing in size
  - Custom rules of more virtual networks
  - QoS demands finer-grained differentiation on rules
  - Increasing number of hosts

- Increasing line-rates
Previous Approach: HiCuts

- Represents rules as cubes in multidimensional space
- Constructs a decision tree by recursively cutting the space and separating rules into different sub-space
- Eventually, rules fall into the leaf nodes
- Upon receiving a packet, the classifier traverses the tree to identify matching rules
Previous Approach: HyperCuts

- Improves upon HiCuts
- Supports multidimensional cutting at tree node
  - Collapse subtrees to reduce tree depth
- Percolates common rules from siblings up to the parent nodes
  - Reduces replication
Memory Overhead of HiCuts and HyperCuts

Varying size of overlapping rules
  ○ Necessary to apply fine cuts for separating the small rules
  ○ Inevitably replicating the large rules

Picture adapted from authors' SIGCOMM presentation
Memory Overhead of HiCuts and HyperCuts

Varying rule-space density
○ Both HiCuts and HyperCuts adopt equi-sized cuts
○ Inadvertently partition sparse space when partitioning dense space
○ Leading to more sub-spaces/tree nodes containing few rules
Optimizations in EffiCuts

- Separable trees
- Selective Tree Merging
- Equi-dense cuts
- Node Co-location
Separable Trees

Intuition: Separate small (fewer wildcards) and large (more wildcards) rules into different trees
- Tree1: \{A, B, C\}, Tree2: \{D, E, F\}

Refinement: A subset of rules are separable if all rules in the subset are either small or large in each dimension
- Tree1: \{A, B, C\}, Tree2: \{D\}, Tree3: \{E, F\}
Selective Tree Merging

- Pitfall of separable trees: more lookups during packet processing and thus lower throughput
- Idea: selectively merge trees
- Complication: merging trees is a compromise on separability
  - Need to minimize replication
  - Merge trees mixing rules that are small or large in at most one dimension
**Equi-dense Cuts**

- Equi-size cuts simplify indexing of matching child but lead to redundancy due to rule-space density variation.

- Equi-dense cuts produce partitions of **similar density** to distribute rules evenly among fewer children by **fusing adjacent equi-sized cuts**.
Node Co-location

- Reduces the amount of memory access
Evaluation

Substantial reduction in memory with modest increase in memory access